Mendez, F., Ferguson, M., Dahnoun, N., & Tancock, S. R. (2018). Real-time user selected dynamic template tracking for UAV. In *2017 IEEE International Conference on Imaging Systems and Techniques (IST 2017)* Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/IST.2017.8261485

Real-Time User Selected Dynamic Template Tracking For UAV

Fabio Mendez
University of Bristol
Email: fm14619@my.bristol.ac.uk

Mark Ferguson
University of Bristol
Email: mf14471@my.bristol.ac.uk

Naim Dahnoun
University of Bristol
Email: naim.dahnoun@bristol.ac.uk

Scott Tancock
University of Bristol
Email: st12660@bristol.ac.uk

***Abstract*—Template matching algorithms describe the way a tracker is able to follow an object by comparing two templates. Representing information about the object to be tracked and the other about the frame that is being analysed. Previous implementations prove to be sensible to partial occlusion, updating the model template even in cases where the line of sight between the camera and the object is being partially blocked, producing a misrepresentation of the object's features. This paper proposes a dynamic template update, using normalised cross correlation as a similarity metric. Using the response given by the similarity, it is possible to determine ranges in which to update the model of the object. With this, the algorithm is able to keep relevant information about the object when it is partially or completely occluded. The main assumption made during the development of most template tracking algorithms is the prior knowledge of the object's location and dimensions in the initial frame. This paper proposes an interactive implementation where the prior information that is needed can be obtained from a single point in the object using segmentation. The implementation of the algorithm described produced reliable and real-time results (30 frames per second) on the NVIDIA Jetson TX1 platform.**

I. Introduction

Object Tracking is a core problem in computer vision, with a wide range of applications which include human-computer interaction, surveillance, augmented reality and performance-driven animation. In some cases this process can be simplified by knowing the object to be tracked in advance, incorporating prior knowledge when designing the tracker to create a model of the object. However, for many applications there is no prior information as the object is specified at run-time. For this scenario the implementation needs to be able to model the object across the duration of the application.

Tracking implementations can be divided into appearance-based and model-based [1]. Appearance-based trackers rely on visual cues (colour, contour, corners, etc.), while model-based trackers make use of a 3D model of the object used for comparison during the video sequence. Implementing a model-based tracker into an UAV can prove to be troublesome, given that in most applications the object to be tracked is specified at run-time. A main requirement for appearance-based trackers is the prior knowledge of the start position and dimensions of the object. This information can be provide by selecting an area of a segmented image that represents the object and bounding this region. The segmented image can be produced by edge based or region growing techniques. Due to the nature of these methods, the whole image is segmented before the area that represents the desired object is considered. A more efficient implementation is to design an interactive object segmentation; for this an input is needed, then based on this the segmentation can be focused on regions of the image the object is most likely to be contained in.

Previous methods to achieve this often require a large amount of user input, either in the form of multiple points [2] or multiple strokes [3] of the object and/or the background. This proves to be problematic for real-time systems, as by the time all the input has been provided, the object could have changed position. To counteract this problem, this paper proposes a method to segment and bound an object based on a single point within it.

Template matching is commonly used to achieve appearance-based tracking. Previous implementations describe the comparison between two templates using a similarity metric such as normalised cross correlation [4] or the sum of absolute value of differences [5]. Other implementations achieve this type of tracking by storing information about the object on each frame and improving a learning algorithm on each iteration [6]. These implementations define a constant model template update, which is problematic, because across the duration of the application, there is a high chance of occlusion, which can lead to misrepresentation of the object. To account for this problem, this paper proposes a dynamic template update, for which the model template is not updated in every iteration. Instead, it defines an update region given the results of the similarity metric used. This not only increases the accuracy of the tracking, but also avoids constant memory writes, improving the algorithm's efficiency. Besides this, an occlusion threshold is also implemented, for which the region of interest (ROI) of the image (see section IV) can be extended, to account for movement of the object while occluded. The block diagram of the proposed tracking system is shown in *Fig. 1*.

This paper is organised as follows: Section II outlines the procedures needed to segment the object from the background and create the tightest bounding box possible covering the object. Sections III, IV and V describe how to use the information from each frame. This includes the usage of different templates and how to extract and match features from them. The following section (section VI) describes how to select the ROI and update of the model template. Section VII describes how the algorithm was implemented and its performance.

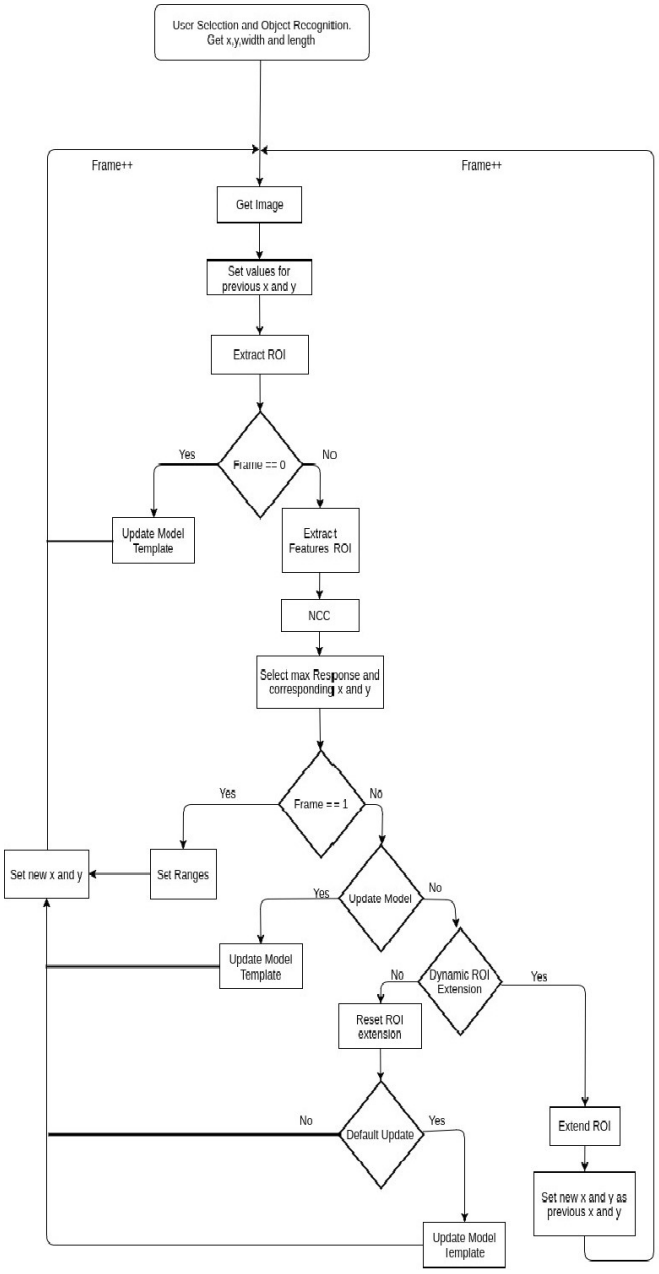

Fig. 1: Tracking algorithm implementation flow chart.

Finally, Section VIII describes improvements that can be made to the procedures described in this paper.

II. Finding Initial Bounding Box

In order to define the object location on the initial frame, a pixel within the object must be provided to extract the segmented object from that area. Initially, the bounding box is defined as the whole image, then at each stage this is examined and the size is reduced where appropriate. The main idea behind this implementation is to incorporate increasingly complex techniques as the bounding box decreases.

Algorithm 1 Finding Bounding Box

function findingBoundingBox(Frame,Input Coordinate)
maxSize(Box,Input Coordinate,currHeight)
backgroundMasking(Box,Mask,Frame)
smooth(Box,Mask)
getClickedRegion(Box,Mask,Input Coordinate)
sizeReduction(Box,Mask,Input Coordinates)
colourSobelBarriers(Box,Mask,Input Coordinates)
getClickedRegion(Box,Mask,Input Coordinates)
return *Box*
end function

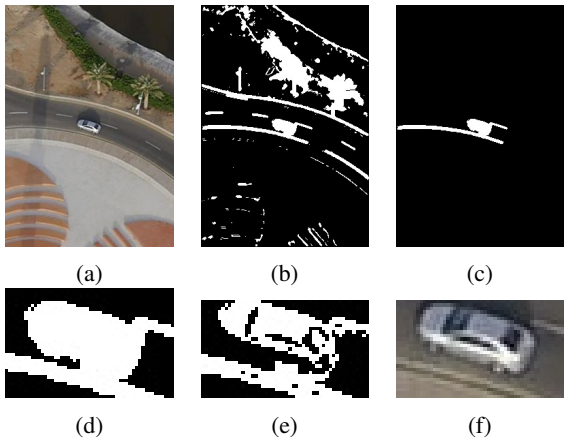

Fig. 2: (a) Input image [7] reduced to bounding box from maxSizeEstimation. (b) Output from smoothing. (c) Output from first getClickedRegion. (d) Output from sizeReduction. (e) Output from second getClickedRegion. (f) Input image reduced to final bounding box.

The pixel width and height of an object can be estimated based on the real object dimensions and the distance between the camera and the floor [8]. So, by assuming a maximum size for the object to be tracked, we can estimate a bounding box covering this area.

Sobel edge detection is applied to obtain the sharp changes in brightness of the image, which in most cases represent the outlines of the object in the image. A threshold is applied to the output of the edge detector. Any pixel greater than this threshold is stored in a binary image as one. Representing an edge, otherwise it is stored as a zero. Connected component analysis (CCA) [9] is then applied to the resulting binary image. This assigns the same label to the pixels that are connected in an object. Given that the initial bounding box is an overestimation, then any pixel in the selected object is not connected outside the initial boundaries. Assuming a fully connected edge was found previously. This means that any region produced by the connected component analysis that touches the edge of the bound box can be deemed as background or a separate object and therefore ignored.

Smoothing is then applied to what so far is considered the object. Disconnecting weakly linked pixels. This is achieved

by a sliding window technique, in which the quantity of pixels labelled as objects inside the window is compared with a threshold. If it is below this, the centre pixel is eliminated. This process is repeated for two different window sizes and thresholds.

CCA is applied again to the output of the smoothing, after which, all the pixels sharing the same label as the input coordinate are extracted.

A recurrent problem with segmentation is under-segmenting, meaning multiple objects are labelled as one. To solve this, colour Sobel barriers are applied, which aim to split objects that have been incorrectly classified. But this implementation is inefficient, so to improve the performance the bounding box is first decreased.

To reduce the dimensions of the bounding box, the system scans horizontally in both directions from the selected point, until it reaches a pixel marked as background on both sides. Measuring the height of the object at each pixel. It scans across before it reaches a pixel labelled as background. This is then averaged to produce a more general result. This process is then repeated vertically to determine the width. However, this reduction of the bounding box is not guaranteed to contain the whole object. To solve this, a multiplier is applied to try and ensure the object is bounded by this area.

Following this, a colour Sobel edge detection is applied [10]. This mask is then subtracted from the bounding box. CCA is applied to this image, and the label containing the initial coordinates is tightly bound and sent to the tracking algorithm.

This implementation is based on the algorithm shown in *Algorithm 1*, as a series of procedures applied to the frame, where the user defines an initial coordinate belonging to the object, and the distance between the camera and the ground is known. Besides this, the outputs of these stages are shown in *Fig. 2*. It can be seen how this implementation is able to tightly bound the selected object using the required input information.

III. Templates

A template is a representation of the important characteristics in a region of the image [5]. The main idea when tracking an object is to compare a model template with several templates created from a region of interest (ROI) of the current frame, and the area from the ROI that presents the highest similarity represents the object.

Two different templates are stored and used by the algorithm. The first one is known as the model and this will hold salient characteristics of the object. The tracking to be explained in this paper presents a dynamic model template generation. This template is updated according to the maximum similarity metric resulting from comparing the object's model with the current templates during the run time of the application. This is done to account for changes in perspective of the object in the camera field of view.

The following assumptions are made when creating the initial model template to ensure an ideal extraction of the features:

- The initial position of the object in relationship to the camera coordinates system is known at the start of the tracking application.
- The object can be completely covered by a bounding box in the image.
- The bounding box dimensions are set such that it doesn't include any other objects.

The second template is known as current and is created from the ROI of the frame to be analysed. Ideally, it will include the characteristics of the object in its new position together with its environment.

To avoid recalculations, the features are initially extracted from the ROI and then divided into sub-sections, based on the dimensions of the object. These subdivisions represent the current templates, and from these the one with the highest similarity score with the model is the new position of the object.

To extract features from the image, the Harris corners edge detection algorithm is used. Generally speaking, corners are areas in an image that present significant variation in densities [11]. This feature extractor is distinctive compared with others because of its simplicity and its robustness to neither rotations nor illumination differences between images [12]. To account for this problem, the dynamic model update proposed in this paper changes the representation of the object for such occasions.

(a) (b)

Fig. 3: (a) Example of an image with (b) corresponding template using Harris corners. Image taken from data set UAV123 [7].

The corners extracted using the specified algorithm are set to have a relatively large block size. With this the features are more general, and the matching implementation produces more reliable results. An example of the feature extraction implemented is shown on *Fig. 3*.

IV. Region of Interest (ROI)

The ROI is defined as the search block in the image. This is a smaller sample of the frame, where the object is most likely to be given its location in the previous frame.

When selecting the location of the ROI, the previous area is extended in all directions creating a bounding box bigger than the object [4]. With this, the probability of the object in its new location being covered by this area tends to unity.

The size of this area greatly impacts the performance (frames per second) of the algorithm. Intuitively, the bigger

this area, the more calculations are needed to detect the similarities of the current templates with the model template. In order to select the smallest extension to create the ROI, the camera frame rate needs to be taken into account. The higher this is, the smaller the extension can be. This phenomenon can be attributed to the fact that if the camera frame rate is high, the object's change of position can be more precisely covered between subsequent frames. This represents a trade-off, between precision and performance.

V. Normalised cross correlation as Template Matching

Normalised cross correlation (NCC) can be defined as a measurement of similarity between two signals. In the implementation defined in this paper, this measurement is used to assign a score comparing the template $T(x', y')$ and several regions in the area of interest $ROI(I(x, y))$. The maximum response defines the new location of the object. NCC calculations are expressed in *Equation 1*.

In the Equations 2 to 5, $T(x, y)$ and $ROI(x, y)$ represents the values of the template image and ROI image respectively in the coordinates x and y. The score obtained using normalised cross correlation is represented in *Fig. 4* as the shaded area for a point (x,y).

$$R(x,y) = \sum_{x'=0}^{lt} \sum_{y'=0}^{bt} \frac{(T(x',y') - \overline{T}) \cdot (I(x+x', y+y') - \overline{I}(x,y))}{\alpha_T \cdot \alpha_I(x,y)} \tag{1}$$

where:

$$\overline{T} = \frac{1}{bt \cdot lt} \sum_{a=0}^{lt} \sum_{b=0}^{bt} T(a,b) \tag{2}$$

$$\overline{I}(x,y) = \frac{1}{br \cdot lr} \sum_{a=0}^{lt} \sum_{b=0}^{bt} ROI(a+x, b+y) \tag{3}$$

$$\alpha_T = \sqrt{\sum_{a=0}^{lt} \sum_{b=0}^{bt} T(a,b)^2} \tag{4}$$

$$\alpha_I(x,y) = \sqrt{\sum_{a=0}^{lt} \sum_{b=0}^{bt} ROI(a+x, b+y)^2} \tag{5}$$

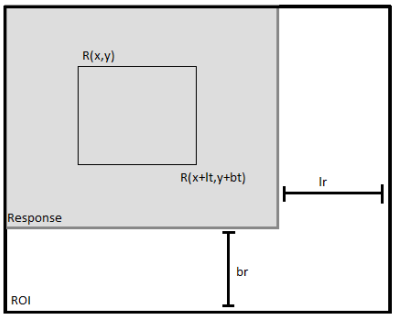

Fig. 4: Example of template matching in area of ROI.

VI. Using Match Metric Results for Model Template Update and Dynamic ROI Selection

The maximum response value returned by the NCC calculations may not represent the new location of the object. This is usually due to partial or complete occlusion of the tracked object. This produces a noticeable decrease of the maximum similarity metric of a frame in relationship with the others. Following this idea, the use of different score areas can be used to define different "action ranges" to find the new object location and the update of the model templates.

In order to decrease memory writes for new model templates, a response range is set which represents an area in the similarity metric which defines when to update the model.

As the first similarity score generally defines a "good match", then there is no need to update the model template for the next iteration. When the defined metric starts to drop, then it can be said that the object is changing its perspective in relationship to the camera and an update is needed to account for this variation.

Constant model updates cause the implementation to become sensitive to partial occlusion. This phenomenon occurs when sections of the tracked object are obstructed leaving others visible for the camera [13]. Given this update area, the model template can retain a good representation of the object even when it is partially occluded.

When updating the model template, a consideration for cases in which the object does not present relevant changes in perspective needs to be made. Or in a worse case scenario, the changes are not completely covered between frames causing the similarity metric to drop considerably between consecutive frames, and with this skipping the model update area. To account for these cases, a default update region is set. This will update the model template in cases where there has not been an updated in a relatively long time.

When the matched metric decreases by a noticeable amount, it can be due to occurrence of an undefined object interfering between the camera line of sight and the tracked object [4], causing complete occlusion. To account for this problem, the new location given by the maximum response is ignored, and the previous location is kept. Besides this, the area of the ROI is extended, in order to account for movement of the object while it was being occluded. After a new match is found with a reliable maximum response, the ROI area is reset to a constant value.

The results of this implementation can be seen in *Fig. 5*. This shows that the tracking algorithm is able to identify occlusion scenarios and keep previous locations of the object. Besides this, *Fig. 6* demonstrates the dynamism on ROI selection for such cases, increasing the search area to account for object's movement while occluded.

VII. Results

One important aspect of this implementation is that the NCC calculations are performed on a GPU using CUDA. The platform chosen for this is an NVIDIA Jetson TX1, mainly because of its size and power consumption (less than 10 watts).

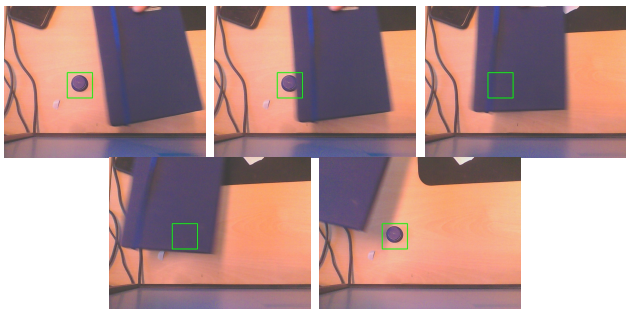

Fig. 5: Example of object being partially and completely occluded

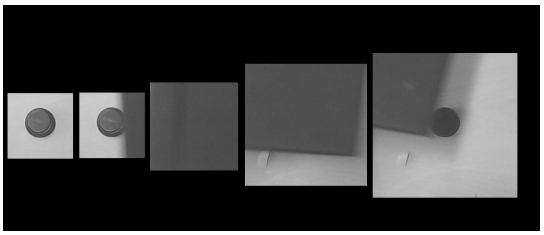

Fig. 6: ROI extension implementation example, showing responses at the moment the object is occluded

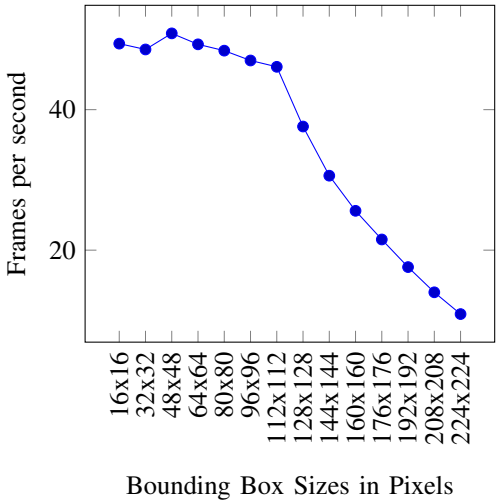

Fig. 7: Scatter plot expressing the resulting frequency of the implementation with different bounding box sizes.

This can easily be incorporated into the UAV, and with this all the calculations can be made on-board. To avoid recalculating the responses for the same frames, the frame rate of the application is locked to be the same as the camera. In this case the camera used is a Creative Model VF0790, which presents a frame rate of 30 frames per second, and the application is able to analyse each frame.

The algorithm described in this paper is able to perform real-time tracking with reliable results up to a bounding box of 144 by 144 pixels. The results shown in *Fig. 7* were obtained when realising tracking on a video given by a data set called UAV123 [7]. This eliminated the restriction set by the camera frame rate, given that on each iteration a new frame was analysed.

An example of the implementation described is shown in *Fig. 8*, clearly demonstrating the effects of the dynamic model template update and ROI selection. The target object presents complete and partial occlusion during the application on this scenario, for which the algorithm is shown to be able to perform a continuous tracking under such circumstances. Besides this, frames 312 to 440 show the effects of the region defined for updating the model template, allowing a constant representation of the object under perspective changes, and avoiding a continuous feature update.

The combination of the algorithms described in this paper permits easy implementation into a UAV, allowing the user to select any object on a video stream, to then be followed by the tracking system. Because of this a wide range of applications can be defined for the procedures described. These include its incorporation with a flight controller for autonomous UAV movements, for which the user is able to define an object for the UAV to follow. Considering the results obtained on the NVIDIA Jetson TX1 platform, this algorithm can be implemented on-board producing reliable and real-time results.

Besides this, the tracking implementation can be incorporated into surveillance systems, in which the user selects an object to be tracked across a video sequence. For this a modification should be made to achieve tracking across images from different sources. Knowing the position of each camera in relationship with each other in the environment, it is possible to create a wider view of the tracking space, allowing continuous tracking with non-mobile video sources.

VIII. Future Work

The feature extraction defined is applied using OpenCV, representing a linear implementation of the Harris corners edge detection. Therefore, one improvement to be made is to create a parallel feature extraction to improve the performance of the algorithm. The sliding window approach used in the Harris corners algorithm allows easy portability for GPU calculations. Besides this, the performance of the method defined can be compared using different parallel feature extraction algorithms across different scenarios. Defining the most efficient and accurate implementation for template matching tracking systems.

Further improvement can be achieved by implementing feature decomposition on the template creation. This is achieved by selecting important regions of the templates that correctly define the target object. Increasing the application's efficiency as the similarity metric is implemented on smaller areas of the templates.

In order to increase the reliability of the tracking system, a variable bounding box needs to be incorporated to cover the object during tracking. The algorithm described presents a constant bounding box, which is able to keep a good representation of the object because of the dynamism in updating the model template. Nevertheless, a more accurate

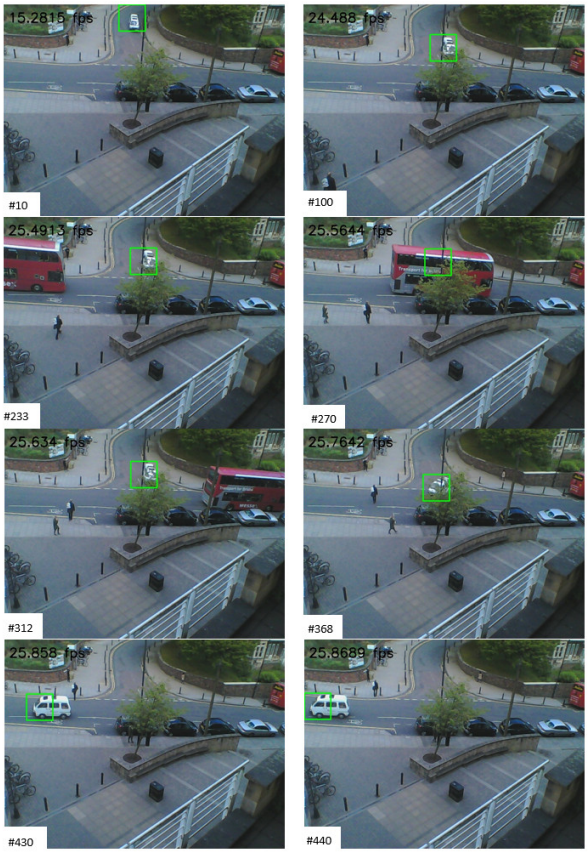

Fig. 8: Example of application being implemented on real tracking scenario. Frame number on video sequence established at the bottom left of each image. Video sequence taken by the authors.

feature extraction to define the model can be achieved by ensuring a tightly bounded box for the object during tracking.

IX. Conclusion

The implementation presented in this paper describes an algorithm which combines object selection and tracking. This yields a complete program, which can be incorporated into an UAV. The idea behind this combination is to cover the several assumptions made by the tracking algorithm.

The segmentation algorithm proposed in this paper segments a single object. The algorithm only examines the image within its current bounding box estimate, allowing more complex computation at latter stages. Increasing the overall performance and accuracy of the segmentation process.

Regarding the tracking implementation, this paper describes a modified template matching implementation, which presents a dynamic template update. This algorithm is able to avoid updating the model template on each iteration. With this, a solution for complete and partial occlusion is presented.

References

[1] J. A. Brown and D. W. Capson, "A framework for 3d model-based visual tracking using a gpu-accelerated particle filter," *IEEE transactions on visualization and computer graphics*, vol. 18, no. 1, pp. 68–80, 2012.

[2] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang, "Deep interactive object selection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 373–381, 2016.

[3] W. Yang, J. Cai, J. Zheng, and J. Luo, "User-friendly interactive image segmentation through unified combinatorial user inputs," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2470–2479, 2010.

[4] S. Majumder and R. Shankar, "Moving object tracking from moving platform," in *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, pp. 85–89, IEEE, 2014.

[5] M. J. Atallah, "Faster image template matching in the sum of the absolute value of differences measure," *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 659–663, 2001.

[6] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[7] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *European Conference on Computer Vision*, pp. 445–461, Springer, 2016.

[8] O. Kainz, F. Jakab, M. W. Horečnỳ, and D. Cymbalák, "Estimating the object size from static 2d image," in *Computing and Communication (IEMCON), 2015 International Conference and Workshop on*, pp. 1–5, IEEE, 2015.

[9] N. Ma, D. G. Bailey, and C. T. Johnston, "Optimised single pass connected components analysis," in *ICECE Technology, 2008. FPT 2008. International Conference on*, pp. 185–192, IEEE, 2008.

[10] X. Chen and H. Chen, "A novel color edge detection algorithm in rgb color space," in *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pp. 793–796, IEEE, 2010.

[11] Y. Li, W. Shi, and A. Liu, "A harris corner detection algorithm for multispectral images based on the correlation," in *6th International Conference on Wireless, Mobile and Multimedia (ICWMMN 2015)*, pp. 161–165, IET, 2015.

[12] M. Roszkowski and G. Pastuszak, "Using harris corner points to reduce the complexity of a local stereo image matching algorithm," in *New Trends in Audio & Video and Signal Processing: Algorithms, Architectures, Arrangements, and Applications (NTAV/SPA), 2012 Joint Conference*, pp. 221–226, IEEE, 2012.

[13] Z. He, S. Yi, Y.-M. Cheung, X. You, and Y. Y. Tang, "Robust object tracking via key patch sparse representation," *IEEE transactions on cybernetics*, vol. 47, no. 2, pp. 354–364, 2017.